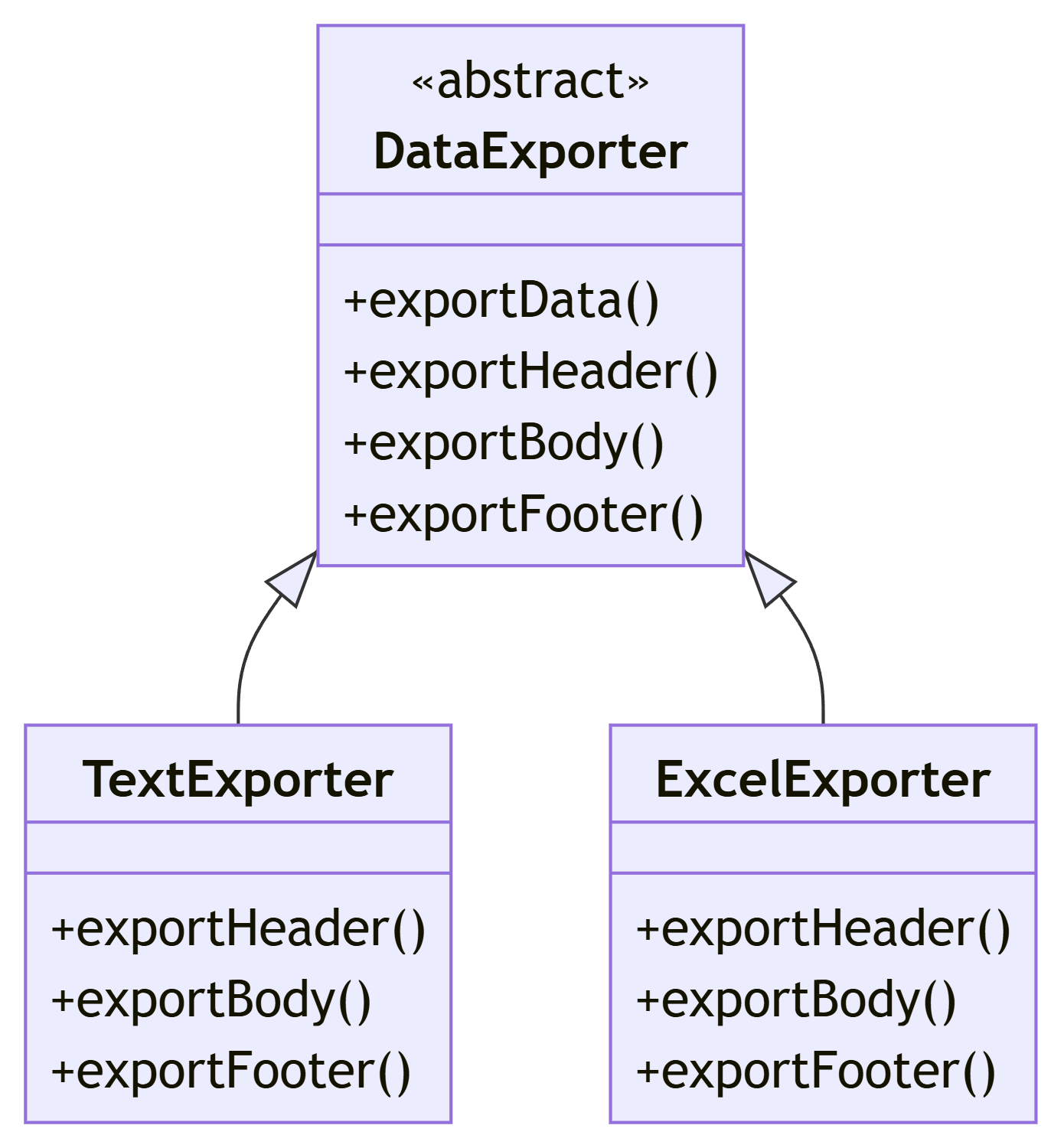
在这个场景中，最合适的设计模式是**模板方法模式**。模板方法模式可以在基类中定义算法的骨架，而将某些具体的步骤推迟到子类中去实现。针对不同的文件格式，比如文本格式、Excel格式等，可以重用相同的导出流程，而具体的文件格式处理逻辑由子类实现。



|  |
| --- |
| **核心代码** |
| #include <iostream>  #include <vector>  #include <string>  #include <ctime>  using namespace std;  // 抽象基类：数据导出器  class DataExporter {  public:  // 模板方法：导出数据，依次调用头、体、尾  void exportData(const string& companyId, const string& tableName, const vector<string>& data, const string& outputPerson) {  exportHeader(companyId);  exportBody(tableName, data);  exportFooter(outputPerson);  }  protected:  // 导出头部，定义为纯虚函数，具体格式由子类实现  virtual void exportHeader(const string& companyId) = 0;    // 导出数据体，定义为纯虚函数，具体格式由子类实现  virtual void exportBody(const string& tableName, const vector<string>& data) = 0;  // 导出尾部，定义为纯虚函数，具体格式由子类实现  virtual void exportFooter(const string& outputPerson) = 0;  // 获取当前日期  string getCurrentDate() {  time\_t now = time(0);  char buf[80];  strftime(buf, sizeof(buf), "%Y-%m-%d", localtime(&now));  return string(buf);  }  };  // 文本格式导出器  class TextExporter : public DataExporter {  protected:  void exportHeader(const string& companyId) override {  cout << "文本格式文件头：公司编号：" << companyId << ", 导出日期：" << getCurrentDate() << endl;  }  void exportBody(const string& tableName, const vector<string>& data) override {  cout << "表名称：" << tableName << endl;  for (const auto& record : data) {  cout << record << endl;  }  }  void exportFooter(const string& outputPerson) override {  cout << "输出人：" << outputPerson << endl;  }  };  // Excel格式导出器（这里只是展示，实际Excel格式处理会复杂得多）  class ExcelExporter : public DataExporter {  protected:  void exportHeader(const string& companyId) override {  cout << "Excel格式文件头：公司编号：" << companyId << ", 导出日期：" << getCurrentDate() << endl;  }  void exportBody(const string& tableName, const vector<string>& data) override {  cout << "表：" << tableName << "（Excel格式）" << endl;  for (const auto& record : data) {  cout << "Excel记录：" << record << endl;  }  }  void exportFooter(const string& outputPerson) override {  cout << "Excel输出人：" << outputPerson << endl;  }  };  // 客户端代码  int main() {  // 模拟导出数据  string companyId = "001";  string tableName = "SalesData";  vector<string> data = {"101, John, 100", "102, Alice, 200"};  string outputPerson = "Admin";  // 创建文本格式导出器  DataExporter\* exporter = new TextExporter();  exporter->exportData(companyId, tableName, data, outputPerson);  cout << endl;  // 创建Excel格式导出器  exporter = new ExcelExporter();  exporter->exportData(companyId, tableName, data, outputPerson);  delete exporter;  return 0;  } |